Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Fluids 33 (2004) 1131–1155

computers & fluids

www.elsevier.com/locate/compfluid

An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids

Eric J. Nielsen [a,*], James Lu [b], Michael A. Park [a], David L. Darmofal [b]

[a] *Computational Modeling and Simulation Branch, MS 128, NASA Langley Research Center, Hampton, VA 23681, USA*

[b] *Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*

Received 7 April 2003; accepted 30 September 2003
Available online 28 February 2004

Abstract

An implicit algorithm for solving the discrete adjoint system based on an unstructured-grid discretization of the Navier–Stokes equations is presented. The method is constructed such that an adjoint solution exactly dual to a direct differentiation approach is recovered at each time step, yielding a convergence rate which is asymptotically equivalent to that of the primal system. The new approach is implemented within a three-dimensional unstructured-grid framework and results are presented for inviscid, laminar, and turbulent flows. Improvements to the baseline solution algorithm, such as line-implicit relaxation and a tight coupling of the turbulence model, are also presented. By storing nearest-neighbor terms in the residual computation, the dual scheme is computationally efficient, while requiring twice the memory of the flow solution. The current implementation allows for multiple right-hand side vectors, enabling simultaneous adjoint solutions for several cost functions or constraints with minimal additional storage requirements, while reducing the solution time compared to serial applications of the adjoint solver. The scheme is expected to have a broad impact on computational problems related to design optimization as well as error estimation and grid adaptation efforts.
© 2004 Elsevier Ltd. All rights reserved.

PACS: 65Y; 76M

* Corresponding author. Tel.: +1-757-864-2239; fax: +1-757-864-8816.
E-mail address: eric.j.nielsen@nasa.gov (E.J. Nielsen).

0045-7930/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2003.09.005

1. Introduction

The field of computational fluid dynamics (CFD) is increasingly important in the design and validation of new aerodynamic concepts. The use of computational tools can greatly reduce the need for more costly alternatives, such as wind tunnel experiments or flight testing. CFD techniques based on the Navier–Stokes equations have matured into reliable tools for the analysis of complex geometries. However, design optimization using these high-fidelity methods has been greatly inhibited by their relatively high expense and lack of robustness.

In an effort to alleviate the high cost of gradient-based design methodologies, recent work has focused on the use of adjoint methods [1–23]. Adjoint techniques generally fall into one of two categories based on the order in which the discretization and differentiation processes are performed. The two approaches are termed continuous or discrete adjoint methods. For a single-output or constraint function, these schemes allow the computation of design sensitivities at the cost of solving a single additional linear problem and a subsequent computation of a matrix–vector product dimensioned by the number of design variables.

In addition to its role in design optimization problems, the solution of the adjoint equation can be used to provide error estimates for an output of engineering interest, as well as grid adaptation information that can be used to improve solution accuracy [24–32]. Traditional methods for grid adaptation have typically relied on ad hoc feature-based quantities [33–36]. In these approaches, features such as shocks, boundary layers, and other regions of interest are characterized by large gradients or curvatures in the solution. The adaptation algorithm then attempts to improve the solution by adding additional grid points in these regions. Unfortunately this approach can yield grids of unwieldy dimensions and even incorrect results [36]. The local flow feature of interest is often over-resolved, while smooth regions of the flowfield are essentially neglected.

The adjoint approach to grid adaptation seeks to minimize the uncertainty or error in some specified output function. In this approach, a local adaptation parameter is obtained by combining flow and adjoint solutions, where the nonlinearities in these solutions are weighted with the local residual error. This adaptation technique implicitly targets the flow features having the highest impact on the output of interest. The test cases examined in [27,28], and [30–32] clearly demonstrate the potential of using such an approach to grid adaptation. Comparisons with feature-based techniques are shown and results equivalent to those of uniform grid-refinement studies are obtained at a fraction of the cost. The process terminates when a user-specified error tolerance is achieved.

Unfortunately the solution of the adjoint system of equations for realistic problems has proven to be a formidable task. In [4], a Krylov method was used to solve the adjoint system for turbulent flows using relatively coarse grids over simple geometries. This work successfully demonstrated the accuracy of the method. However, the computational time required to solve the equations was as much as ten times the cost of the analysis problem, and the scheme failed to converge for many problems. By employing a more extensive preconditioner for the Krylov algorithm, [5] demonstrated improved performance. However, this preconditioning strategy was shown to require approximately five times the memory of the baseline analysis scheme. This approach proved infeasible on available computers for large-scale problems that require grids containing several million mesh points.

The focus of the current work is to develop a new solution algorithm for the discrete adjoint system described in [4,5]. The work is largely based on the recent contributions of Giles, [20–23] in which adjoint solutions for the Euler and Navier–Stokes equations are computed by using an explicit Runge–Kutta scheme combined with multigrid using an exact dual method. Elliott [37] recently used a similar approach for two-dimensional turbulent flows on overset structured grids using multigrid and approximate factorization.

Improvements to the implicit solution technique of [38,39] are described, and an exact dual scheme is developed for the resulting algorithm. Care has been taken to ensure that the implementation yields identical values for linear functionals at each time step, thereby guaranteeing identical asymptotic convergence rates for the primal and dual systems. Results are shown for several small test problems and two large-scale configurations. Convergence of a typical cost function and its derivatives are examined, and computational speed and memory requirements are discussed. The efficiency gained through the simultaneous computation of adjoint solutions for several output functions is also presented.

2. Flow solution method

The governing equations are the three-dimensional Reynolds-averaged Navier–Stokes equations. For turbulent flows, the one-equation model of Spalart and Allmaras [41] is used. The flow solver used in the current work is described at length in [1,38,39]. The code uses an implicit, upwind, finite-volume discretization in which the dependent variables are stored at the mesh vertices. Inviscid fluxes at cell interfaces are computed by using the upwind schemes of Roe [42] or Van Leer [43]. Viscous fluxes are formed by using an approach equivalent to a central-difference Galerkin procedure. For steady-state flows, temporal discretization is performed by using a backward-Euler time-stepping scheme. A highly scalable parallelization scheme is achieved through domain decomposition and MPI communication.

An approximate solution of the linear system of equations formed at each time step is obtained through several iterations of a point-iterative scheme in which the nodes are updated in an even–odd fashion, resulting in a Gauss–Seidel-type method. This scheme is augmented with a line-relaxation algorithm in the current work.

In [1–5,38,39], the turbulence model is integrated all the way to the wall without the use of wall functions, and is solved separately from the flow equations at each time step with an identical time-stepping scheme. The resulting linear system is solved with the same iterative scheme employed for the flow equations. The impact of coupling the flow equations and turbulence model will be addressed further in a subsequent section.

In [1,4,5], a discrete adjoint capability has been developed for the solver. In these references, the discretization of the flow equations and turbulence model described above has been fully differentiated by hand, and the adjoint system of equations has been solved by using a preconditioned GMRES [40] algorithm. The focus of the current work is an alternate solution method based on an exact dual formulation.

2.1. Tightly coupled turbulence model

The loosely coupled implementation of the turbulence model in the baseline solver was originally chosen for its convenience in accommodating additional models as they became available.

Furthermore, the loose formulation allows for straightforward implicit enforcement of positivity on the eddy viscosity by using M-type matrices and positive operators as described in [41,44].

Although the loose formulation has proven satisfactory for engineering-level analysis, it often results in stalled convergence or limit-cycle oscillations that can be detrimental to subsequent adjoint computations. In the current work, a tightly coupled algorithm is used to obtain more robust convergence behavior. The scheme includes the linearizations of the governing flow equations with respect to the turbulence model dependent variable $\tilde{\upsilon}$, as well as the linearizations of the turbulence model with respect to the conserved variables Q, in the computation of the solution updates ΔQ and $\Delta\tilde{\upsilon}$. The approach taken in [41,44] to guarantee positivity on $\tilde{\upsilon}$ becomes prohibitively difficult to impose for the tightly coupled system, so that an update clip at $\tilde{\upsilon} = 0$ is necessary to preclude nonphysical behavior as the solution rapidly develops from its initial freestream conditions. This procedure occasionally admits transients in the early stages of a computation which may result in divergence of the solution; these transients are overcome by using small time steps as the initial solution sets up. Although not discussed in the present paper, limited success has been achieved through modifications as suggested in [45], in which an analysis of the linearized form of the turbulence source terms suggests a similar addition to the diagonal elements of the system.

To demonstrate the effect of coupling on solution convergence, transonic flow over an ONERA M6 wing [46] is computed by using both the loosely and tightly coupled formulations. The grid is shown in Fig. 1 and contains 359,536 nodes and 2,074,955 tetrahedra. The freestream Mach number is 0.84, the angle of attack is 3.06°, and the Reynolds number is 5 million based on the mean aerodynamic chord. For both cases, the CFL number for the flow and turbulence equations has been linearly ramped from 10 to 200 over the first 50 iterations. The convergence histories for density and turbulence for the two solution methods are shown in Fig. 2. The loosely coupled scheme results in stalled convergence, whereas the tightly coupled scheme steadily reduces the

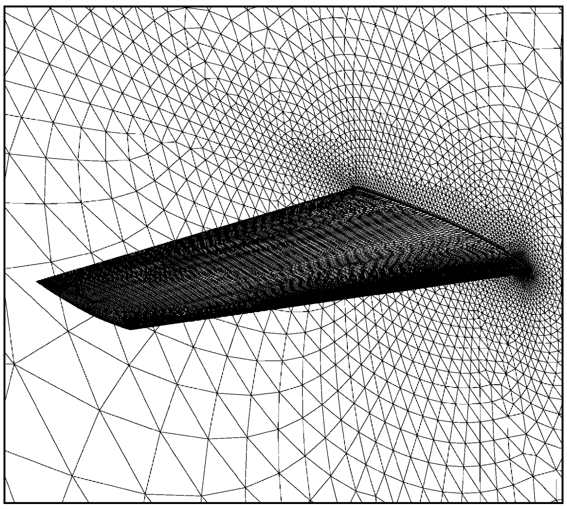
Fig. 1. Surface grid for viscous ONERA M6 wing.

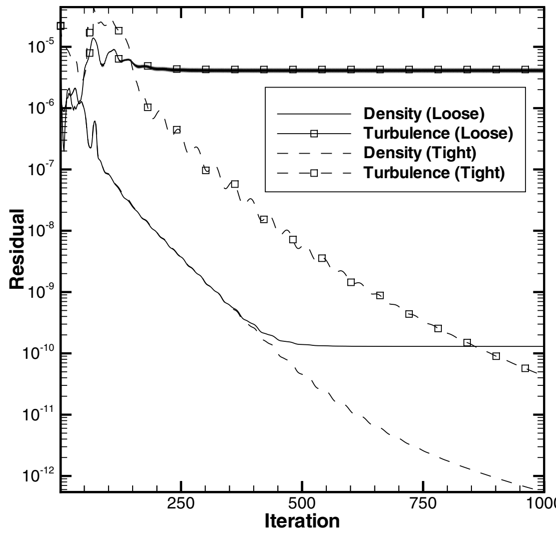

Fig. 2. Effect of turbulence model coupling on ONERA M6 flow solution.

residuals by six orders of magnitude over the course of the computation. Similar results have been observed in other cases, and the tightly coupled algorithm is employed for the remainder of the current work.

2.2. Line-implicit relaxation scheme

When used in conjunction with the baseline point-implicit scheme used for the flow solution, a line-implicit scheme can result in improved convergence rates for viscous flows. The benefits of line-relaxation techniques on highly stretched grids are well-known [47–49]. The central idea is that the coupling of the discrete equations is considerably higher in the direction normal to the grid stretching, so that a relaxation scheme can be constructed to more efficiently solve the equations associated with lines in these directions.

2.2.1. Line construction method

On structured grids the line-relaxation direction is typically well-defined because a grid coordinate direction can be readily employed. For unstructured grids, this inherent direction is not available and must be constructed prior to performing any computations. In [49], implicit lines have been constructed based on neighboring prismatic and hexahedral elements within the boundary layer. In the current work, only tetrahedral elements are employed; thus, an alternative line construction strategy must be used.

To construct a line for implicit relaxation through the boundary layer, an initial point is chosen on a viscous boundary surface. A surface normal is constructed at this point, and the direction of edges connected to this node are compared with the normal using an inner product. The edge with the maximum positive inner product is selected to form the first line segment. The surface normal is then replaced with the direction of the chosen edge, and the process is repeated at the endpoint of successively selected edges, forming the line along which to relax. The process is terminated when the ratio of the length of the longest edge connected to the current node to that of the selected edge falls below a predefined value; a value of 5 is used in the current work. This criterion serves to identify the "inviscid" region of the grid, where elements revert to an isotropic distribution. Construction of a line is also terminated if an edge direction differing by less than 20° from the previous segment cannot be found. This second stopping criterion ensures that line segments remain normal to the original boundary surface. The result of applying this line construction algorithm to a simple wingtip geometry [50] is shown in Fig. 3. A direction suitable for line-implicit relaxation through the boundary layer has been formed at each boundary node, such that approximately 30 grid points are contained in each line.

Another important consideration is the mesh partitioning strategy for parallel processing. To efficiently perform relaxation along any given line, the line should wholly lie within a grid partition. To avoid partition boundaries cutting across relaxation lines, edge weighting is employed in the partitioning phase [51]. This procedure minimizes the number of implicit lines that are split across partition boundaries. In the event that an implicit line is cut by the mesh partitioning, the line is terminated at the partition boundary; no attempt is made to perform line-relaxation across processors. Finally, if the edge weighting scheme is the sole constraint provided to the partitioning algorithm, a partitioning will be generated with an equidistributed set of nodes that largely satisfies this constraint; however, there will be no guarantee that the line- and point-implicit regions themselves will be load-balanced. For this reason, additional node-based weights are provided to a multiconstraint version of the partitioning algorithm [52] to ensure that load balancing is also achieved within each relaxation region.

2.2.2. Line-relaxation algorithm

Once the linear system of equations at the current time step has been assembled, the unknowns associated with each implicit line are computed exactly by using Gaussian elimination of

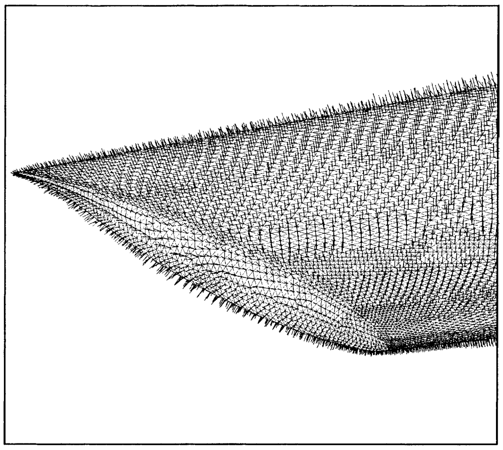

Fig. 3. Result of implicit line construction algorithm applied to wingtip geometry of [50].

a local block-tridiagonal system of equations. This procedure is repeated for a series of sweeps over the implicit lines; the initial decomposition of the coefficient matrix is stored so that subsequent sweeps merely consist of a forward/backward substitution procedure. Once a suitable level of convergence is obtained for the unknowns, the remainder of the domain is relaxed by using several sweeps of the baseline point-implicit scheme; adjacent unknowns determined by the line-implicit scheme are taken as known quantities and moved to the right-hand side of the equations.

To demonstrate the benefits of the line-implicit scheme, fully turbulent flow over the wing-body configuration [50] shown in Fig. 4 is computed on twenty-two 2.2 GHz Pentium IV processors using the baseline point-implicit scheme as well as the line-implicit algorithm. The freestream Mach number is 0.75, the angle of attack is 0°, and the Reynolds number is 3 million based on the mean aerodynamic chord. The grid contains 1,641,452 nodes and 9,650,684 tetrahedra, and the line construction algorithm places 1,069,238 nodes in the line-implicit region. For this test, 15 sweeps through the line- and point-implicit regions are used for both computations.

Convergence of the density and turbulence residuals for both solution schemes is shown in Fig. 5. For this case, a small region of separation near the trailing-edge wing-body juncture prevents monotonic convergence behavior for both schemes, which can be observed in the latter stages of the line-implicit results. However, the line-implicit strategy ultimately results in a computational savings of approximately 20% over the baseline algorithm. As shown in Fig. 6, lift and drag more rapidly approach their steady-state values with the line-implicit scheme. This behavior is attributed to the rapid development of the boundary layer region, and has been observed in a number of cases. For comparison, the results are plotted in Fig. 7 against experimental values from [50]. Similar to many of the computations reported in [53], the lift is slightly overpredicted; however, the correlation with the experimental drag polar is good.

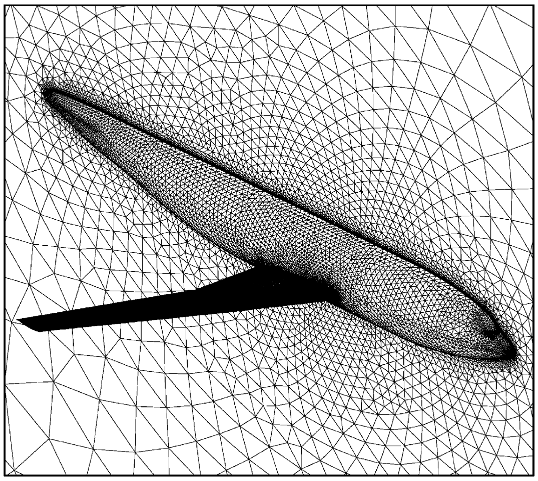

Fig. 4. Surface grid for configuration of [50].

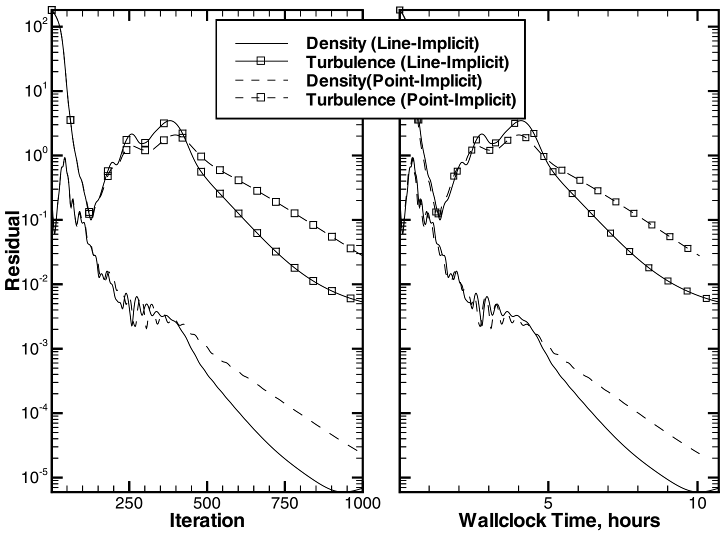

Fig. 5. Residual convergence histories for point- and line-implicit solutions.

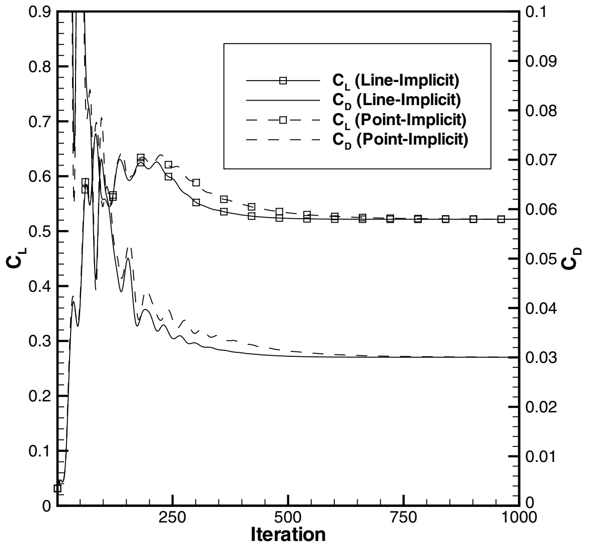

Fig. 6. Force convergence for point- and line-implicit solutions.

To examine the effect of grid refinement, fully turbulent computations are performed on a family of three grids for the geometry shown in Fig. 1 using twenty-four 2.6 GHz Pentium IV processors. The coarse, medium, and fine grids contain 38,823, 297,208, and 2,324,725 nodes and 224,169, 1,752,560, and 13,852,408 tetrahedra, respectively, and have been generated using a global refinement procedure based on the coarse grid. The line construction algorithm results in

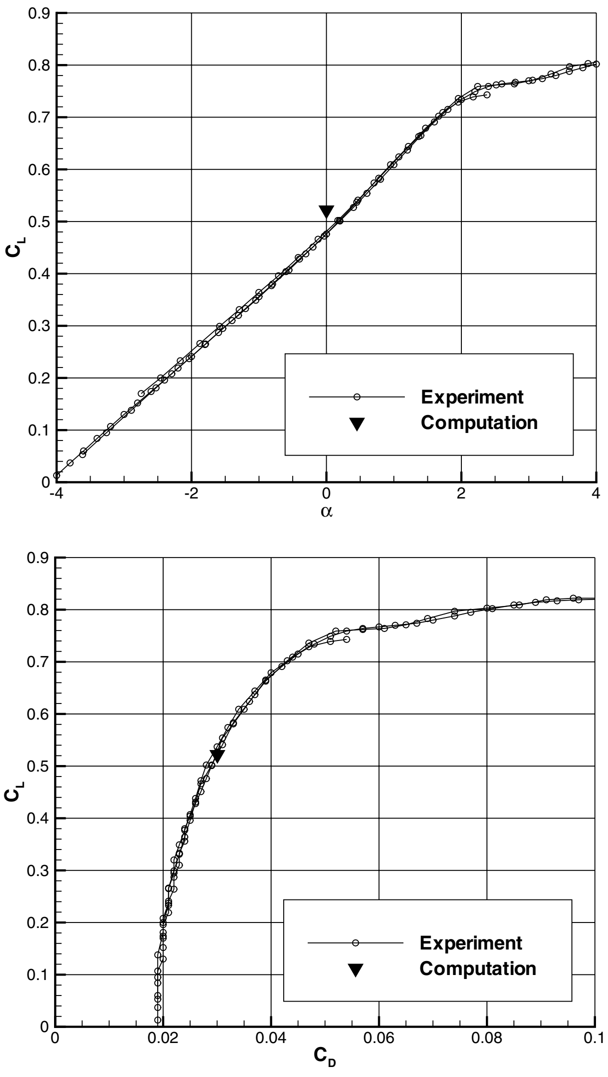

Fig. 7. Comparison of computed force coefficients with experimental data of [50].

23,863, 175,229, and 1,336,656 nodes being placed in the line-implicit region for each successive grid level. For these cases, the freestream conditions are identical to those used for the results shown in Fig. 2, and each computation employs ten sweeps through the linear systems at each time step. The density residuals for each solution are shown in Fig. 8. The turbulence residual has been found to converge in a comparable manner for each case and is omitted here for clarity. The two relaxation schemes yield similar convergence behavior on the coarse and medium grids; however, the performance of the point-implicit algorithm deteriorates considerably on the fine grid.

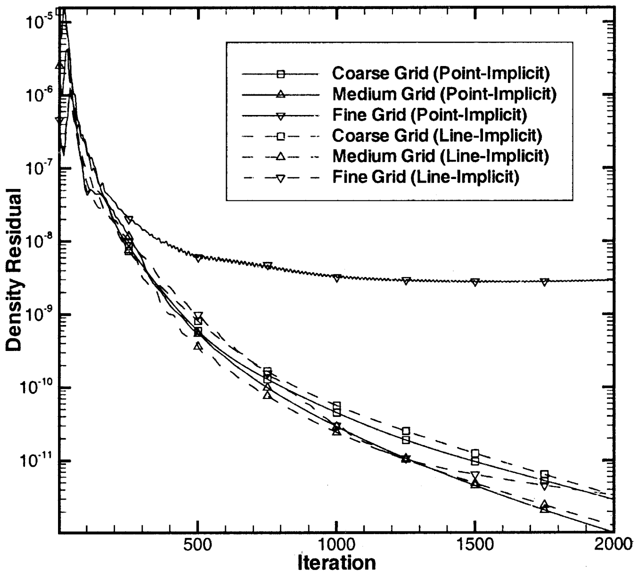

Fig. 8. Effect of grid refinement on point- and line-implicit ONERA M6 flow solutions.

3. Derivation of the dual algorithm

The exact dual algorithm is based on the algorithm used to solve the nonlinear primal equations for the flow unknowns. Consider the following form of the steady-state nonlinear governing equations, where $\boldsymbol{D}$ and $\boldsymbol{Q}$ represent the vector of design variables and the corresponding dependent variables, respectively, and $\boldsymbol{R}_2$ represents a second-order accurate discretization of the spatial residual.

$$\boldsymbol{R}_2(\boldsymbol{Q},\boldsymbol{D}) = 0. \tag{1}$$

Note that in practice, there is also an implicit dependency on the computational grid; the current approach includes these terms, however they are omitted here to simplify the underlying analysis.

An iterative algorithm based on a backward-Euler integration scheme for the solution of the nonlinear system of equations given by Eq. (1) can be written as

$$\frac{V}{\Delta t}\boldsymbol{I}(\boldsymbol{Q}^{n+1} - \boldsymbol{Q}^n) + \boldsymbol{R}_2(\boldsymbol{Q}^{n+1},\boldsymbol{D}) = 0, \tag{2}$$

where V and Δt represent the local cell volume and time step, respectively, and $\boldsymbol{Q}^n$ is the vector of dependent variables at time step n. The nonlinear iterative scheme is attained by linearizing the residual about time step:

$$\left(\frac{V}{\Delta t}\boldsymbol{I} + \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}}\right)\Delta \boldsymbol{Q}^n + \boldsymbol{R}_2(\boldsymbol{Q}^n,\boldsymbol{D}) = 0, \tag{3}$$

where $\Delta \boldsymbol{Q}^n = \boldsymbol{Q}^{n+1} - \boldsymbol{Q}^n$. Note the iteration superscript n is omitted from the Jacobian matrix $\partial \boldsymbol{R}_2/\partial \boldsymbol{Q}$ for clarity; unless otherwise noted, these derivatives will all be evaluated at time step n.

Since the Jacobian used in Eq. (3) is an exact linearization of $\boldsymbol{R}_2$, then for an infinite time step Δt, the iterative scheme corresponds to Newton's method, and $\boldsymbol{Q}^{n+1}$ will converge quadratically to the steady-state value $\boldsymbol{Q}^*$ corresponding to $\boldsymbol{D}$. However, the use of an exact Jacobian $\partial \boldsymbol{R}_2/\partial \boldsymbol{Q}$ is often prohibitively expensive for realistic problems, so that an approximate form of Eq. (3) is typically used:

$$\left(\frac{V}{\Delta t}\boldsymbol{I} + \frac{\partial \boldsymbol{R}_1}{\partial \boldsymbol{Q}}\right)\Delta \boldsymbol{Q}^n + \boldsymbol{R}_2(\boldsymbol{Q}^n, \boldsymbol{D}) = 0, \tag{4}$$

where the exact Jacobian has been replaced with a linearization of some first-order approximation to the spatial residual. The drawback to such an approach is that the asymptotic convergence of the algorithm becomes linear in nature.

Eq. (4) is a linear system of equations for ΔQ^n, which in principle can be solved exactly. In practice, however, the system is usually solved approximately by using an iterative method. Therefore, Eq. (4) can be restated as

$$\Delta Q^n + \boldsymbol{P}\boldsymbol{R}_2(\boldsymbol{Q}^n, \boldsymbol{D}) = 0, \tag{5}$$

where $\boldsymbol{P}$ is an approximation to the inverse of $V/\Delta t\boldsymbol{I} + \partial \boldsymbol{R}_1/\partial \boldsymbol{Q}$, typically achieved through some iterative scheme such as Gauss–Seidel. In this manner, the rate of convergence is governed by the largest eigenvalue of the matrix $\boldsymbol{I} - \partial(\boldsymbol{P}\boldsymbol{R}_2)/\partial \boldsymbol{Q}$. As $\boldsymbol{R}_2$ approaches zero, the asymptotic rate of convergence is then determined by the largest eigenvalue of the matrix $\boldsymbol{I} - \boldsymbol{P}\partial \boldsymbol{R}_2/\partial \boldsymbol{Q}$.

3.1. Direct differentiation algorithm

With minor modifications, the iterative algorithm outlined above can be used to determine the sensitivity derivatives of an output function $f = f(\boldsymbol{Q}^*, \boldsymbol{D})$ with respect to a design variable. Application of the chain rule yields the following:

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{D}} = \frac{\partial f \partial \boldsymbol{Q}^*}{\partial \boldsymbol{Q}^* \partial \boldsymbol{D}} + \frac{\partial f}{\partial \boldsymbol{D}}. \tag{6}$$

Linearizing Eq. (1) with respect to $\boldsymbol{D}$ gives the linear residual equations for the sensitivity derivatives $\partial \boldsymbol{Q}^*/\partial \boldsymbol{D}$:

$$\frac{\partial \boldsymbol{R}_2 \partial \boldsymbol{Q}^*}{\partial \boldsymbol{Q}^* \partial \boldsymbol{D}} + \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}} = 0. \tag{7}$$

Applying the algorithm of Eq. (3) to this expression gives an iterative scheme for $\partial \boldsymbol{Q}^*/\partial \boldsymbol{D}$:

$$\left(\frac{V}{\Delta t}\boldsymbol{I} + \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}}\right)\Delta\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n + \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n + \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}} = 0. \tag{8}$$

Finally, replacing the exact Jacobian with the approximate Jacobian as before and performing an approximate inverse gives the final form of the direct differentiation iterative algorithm for the sensitivity derivatives $\partial \boldsymbol{Q}^*/\partial \boldsymbol{D}$:

$$\Delta\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n + \boldsymbol{P}\left[\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n + \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}}\right] = 0. \tag{9}$$

Note the asymptotic rate of convergence is again dictated by the largest eigenvalue of $\boldsymbol{I} - \boldsymbol{P}\partial\boldsymbol{R}_2/\partial\boldsymbol{Q}$, and therefore will be equivalent to that of the scheme used to attain $\boldsymbol{Q}^*$. After N iterations of Eq. (9), the derivative of the output f with respect to the design variables $\boldsymbol{D}$ may be approximated as

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{D}} = \frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N + \frac{\partial f}{\partial \boldsymbol{D}}. \tag{10}$$

3.2. Exact dual algorithm

In this section, an iterative solution of the dual problem is constructed in a manner which guarantees that the functional estimate is identical at every iteration to that obtained from the direct differentiation algorithm. Using an adjoint approach, [3,19] the sensitivity derivative of the output can be written as

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{D}} = -\Lambda^{\mathrm{T}}\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}} + \frac{\partial f}{\partial \boldsymbol{D}}, \tag{11}$$

where Λ is an adjoint variable that satisfies

$$\left[\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\right]^{\mathrm{T}}\Lambda - \left[\frac{\partial f}{\partial \boldsymbol{Q}^*}\right]^{\mathrm{T}} = 0. \tag{12}$$

Since Eq. (12) is independent of the vector $\boldsymbol{D}$, the adjoint approach is particularly attractive for aerodynamic design problems with a large number of design variables and relatively few constraints. By multiplying Eq. (7) by Λ^{T} and substituting Eq. (12), it can be show that

$$\frac{\partial f \partial \boldsymbol{Q}^*}{\partial \boldsymbol{Q}^* \partial \boldsymbol{D}} = -\Lambda^{\mathrm{T}}\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}}. \tag{13}$$

Therefore, the values of $\mathrm{d}f/\mathrm{d}\boldsymbol{D}$ calculated from Eqs. (6) and (11) are identical. However, since an iterative algorithm is used to estimate $\partial\boldsymbol{Q}^*/\partial\boldsymbol{D}$, this relationship will not hold in general. Following Giles' exact dual approach, an iterative adjoint solution algorithm is sought which satisfies

$$\frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N = -(\Lambda^N)^{\mathrm{T}}\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}}, \tag{14}$$

where N is some iteration at which the equality is to be enforced. However, the resulting exact dual algorithm will not depend on N, and the expression given by Eq. (14) will be valid at every iteration. In this sense, the adjoint iterative algorithm will be exactly dual to the direct differentiation algorithm.

Introducing Lagrange multipliers, the left-hand side of Eq. (14) after N iterations can be written as

$$\frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N = \frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N - \sum_{n=0}^{N-1}(\lambda^{n+1})^{\mathrm{T}}\left\{\Delta\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n + \boldsymbol{P}\left[\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n + \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}}\right]\right\}. \tag{15}$$

Then, by defining

$$\Lambda^n = \sum_{m=N-n}^{N-1} \boldsymbol{P}^{\mathrm{T}} \lambda^{m+1}, \tag{16}$$

and performing the manipulations shown in Appendix A, the exact duality condition requires that

$$\Lambda^{n+1} - \Lambda^n + \boldsymbol{P}^{\mathrm{T}}\left[\left(\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\right)^{\mathrm{T}} \Lambda^n - \left(\frac{\partial f}{\partial \boldsymbol{Q}^*}\right)^{\mathrm{T}}\right] = 0 \tag{17}$$

with an initial condition $\Lambda^0 = 0$. As mentioned previously, Eq. (17) is independent of N and thus the exact duality condition holds for all values of N. This condition guarantees an asymptotic rate of convergence for the dual problem which is governed by the largest eigenvalue of $\boldsymbol{I} - \boldsymbol{P}^{\mathrm{T}}[\partial \boldsymbol{R}_2/\partial \boldsymbol{Q}^*]^{\mathrm{T}}$ and is therefore ultimately equivalent to that of the solution for $\boldsymbol{Q}^*$.

3.2.1. Relationship between $\boldsymbol{P}$ and $\boldsymbol{P}^T$

In the current work, the approximate inverse $\boldsymbol{P}$ is the result of applying multiple iterations of a fixed-point method to a system of the form $\boldsymbol{Ax} = \boldsymbol{b}$. Here, the fixed-point method is given by the line- and point-implicit schemes described above, which can be written in the following form:

$$\boldsymbol{Mx}^{j+1} = \boldsymbol{b} + \boldsymbol{Nx}^j \tag{18}$$

or

$$\boldsymbol{x}^{j+1} - \boldsymbol{x}^j = \boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{Ax}^j), \tag{19}$$

where $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$. Here, $\boldsymbol{M}$ is some matrix that is easily invertible and approximates $\boldsymbol{A}$.

Given some initial estimate $\boldsymbol{x}^0$, after J iterations the scheme results in the following:

$$\boldsymbol{x}^J - \boldsymbol{x}^0 = [\boldsymbol{I} - (\boldsymbol{M}^{-1}\boldsymbol{N})^J A^{-1}](\boldsymbol{b} - \boldsymbol{Ax}^0). \tag{20}$$

Therefore, with respect to Eq. (9), the approximate inverse $\boldsymbol{P}$ using $\boldsymbol{J}$ iterations of the fixed-point method is

$$\boldsymbol{P} = \boldsymbol{I} - (\boldsymbol{M}^{-1}\boldsymbol{N})^J \boldsymbol{A}^{-1}. \tag{21}$$

Forming the transpose of $\boldsymbol{P}$ yields

$$\boldsymbol{P}^{\mathrm{T}} = \boldsymbol{I} - \boldsymbol{A}^{-\mathrm{T}}[(\boldsymbol{M}^{-1}\boldsymbol{N})^{\mathrm{T}}]^J. \tag{22}$$

Because $\boldsymbol{M}^{-1}\boldsymbol{N} = \boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}$, then

$$\boldsymbol{P}^{\mathrm{T}} = \boldsymbol{I} - \boldsymbol{A}^{-\mathrm{T}}[(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})^{\mathrm{T}}]^J$$

or

$$\boldsymbol{P}^{\mathrm{T}} = \boldsymbol{I} - [\boldsymbol{M}^{-\mathrm{T}}\boldsymbol{N}^{\mathrm{T}}]^J \boldsymbol{A}^{-\mathrm{T}}. \tag{23}$$

If the dual problem takes the form $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} = \boldsymbol{g}$ as in Eq. (17), then analogous to Eq. (20), Eq. (23) implies that the fixed-point iterative scheme for the dual problem is

$$\boldsymbol{y}^{j+1} - \boldsymbol{y}^{j} = \boldsymbol{M}^{-\mathrm{T}}(\boldsymbol{g} - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}^{j}). \tag{24}$$

For a point-Jacobi scheme, the transpose operator applied to the iteration matrix $\boldsymbol{M}$ is of no consequence, other than to imply that the diagonal blocks are to be transposed. However, for a Gauss–Seidel iterative method such as the one used in the current work, the transpose operation transforms an upper-triangular iteration matrix into a lower-triangular form, and vice versa. Therefore, the transpose implies that even–odd sweeps through the system of equations must be performed in a reverse manner. The same argument holds for the line-implicit operator used to relax the boundary layer region, although the solution within each line can be obtained in the same manner as the baseline scheme because the inversion is exact for the set of local equations within the line.

4. Implementation of the dual algorithm

The majority of the effort associated with developing an adjoint solver is in forming the exact linearizations of the second-order residual. For the codes used in the current study, this differentiation has been previously performed and the accuracy has been established in [1,3,5]. The implementation of the exact dual algorithm is primarily a high-level task that involves the manipulation of existing routines; this process is made easier through the programming practices established in [54].

After the flow solution has completed a series of n time steps using the algorithm given by Eq. (5), the adjoint solve can be performed in an analogous fashion according to Eq. (17). Note that the dual iterative scheme is essentially identical to the baseline analysis scheme so that little extra coding is required.

According to Eq. (17), the exact linearization of the second-order spatial residual is required for the adjoint computation. However, since these terms only appear in a matrix–vector product, they can be formed on a term-by-term basis and therefore do not require the extensive storage typically associated with the higher-order stencil. The transpose of the first-order Jacobian matrix $(V/\Delta t)\boldsymbol{I} + \partial\boldsymbol{R}_1/\partial\boldsymbol{Q}^*$ is used to solve the system of adjoint equations, so that the storage required by the exact dual scheme is roughly equivalent to that of the baseline analysis code. This requirement is in contrast to the solution algorithm outlined in [5], in which the complete linearization of the second-order residual $[\partial\boldsymbol{R}_2/\partial\boldsymbol{Q}^*]^{\mathrm{T}}$ was used as a preconditioner. The terms arising from this larger stencil resulted in a memory requirement approximately five times that of the scheme outlined in [38,39]. Finally, since the Jacobian matrix $\partial\boldsymbol{R}_1/\partial\boldsymbol{Q}^*$ is constant for an adjoint computation, the block and tridiagonal-block decompositions used for the point- and line-implicit iterations need only be performed once and stored.

The solution strategy outlined in [1] and [4] required a frequent computation of the form $[\partial\boldsymbol{R}_2/\partial\boldsymbol{Q}^*]^{\mathrm{T}}\Delta\Lambda$ for the Krylov method being used. The viscous terms arising from the nearest-neighbor discretization were stored, so that only the inviscid terms involving the larger stencil were recomputed for each new Krylov vector. In the current work, an analogous approach can be used in the formation of the adjoint residual component $[\partial\boldsymbol{R}_2/\partial\boldsymbol{Q}^*]^{\mathrm{T}}\Lambda^n$ if additional memory is available. Furthermore, if sufficient memory is available for the higher-order terms as used for the preconditioner in [5], the complete linearization may be stored. In this manner, the residual update at time step n reduces to an explicit matrix–vector product. Since all terms comprising the residual are stored in this approach, the solution time for the adjoint problem could be reduced considerably. Solution times and memory requirements for each of these strategies will be shown in a subsequent section.

To alleviate the expense associated with computing further adjoint solutions when design sensitivities or mesh adaptation criteria for additional output functions are desired, the current implementation allows for multiple right-hand side vectors to be used. Since the vectors Λ, $\Delta\Lambda$, and the associated residual are the only terms that depend on each output function f, the amount of additional storage required for this approach is minimal. A separate computation of the quantities $[\partial \boldsymbol{R}_2/\partial \boldsymbol{Q}^*]^{\mathrm{T}}\Lambda^n$ and $\Delta\Lambda^n$ is necessary for each f; however, the linearization of $\boldsymbol{R}_2$ at each iteration, the initial block- and block-tridiagonal decompositions of the coefficient matrix, and the additional overhead of initializing and performing a computation need only be performed once. The efficiency of this approach will be demonstrated in a subsequent section.

5. Validation cases

A series of small test cases involving inviscid, laminar, and turbulent flow over the ONERA M6 wing geometry is presented to verify that the exact dual algorithm has been implemented correctly. Each of the viscous tests has been run using the line-implicit relaxation scheme. For each example, the property of Eq. (14) has been verified to machine accuracy, and the asymptotic convergence rates for density (and turbulence where appropriate) are shown to be equal to that of the corresponding adjoint equations. For visual clarity, the convergence histories of the momentum and energy equations and their adjoint counterparts are not shown in the figures that follow, but have been found to closely fellow that of the density equation. The cost function for each test case is based on lift, and the CFL number for the flow solution has been linearly ramped from 10 to 200 over the first 50 iterations. Unless otherwise stated, each of the computations shown here has been performed on sixteen 2.2 GHz Pentium IV processors.

5.1. Inviscid wing

The first test case is inviscid flow, with a freestream Mach number equal to 0.84 and an angle of attack of 3.06°. The grid for this test contains 357,900 nodes and 2,000,034 tetrahedra. The convergence histories of the density equation for the flow and adjoint solutions are shown in Fig. 9; the convergence rates are equivalent.

5.2. Laminar wing

The first viscous case is for laminar flow with a freestream Mach number of 0.3, an angle of attack of 2° and a Reynolds number of 1000 based on the mean aerodynamic chord. The grid shown in Fig. 1 is used for this example. The density convergence histories for the flow and adjoint solutions are shown in Fig. 9, and the convergence rates are identical.

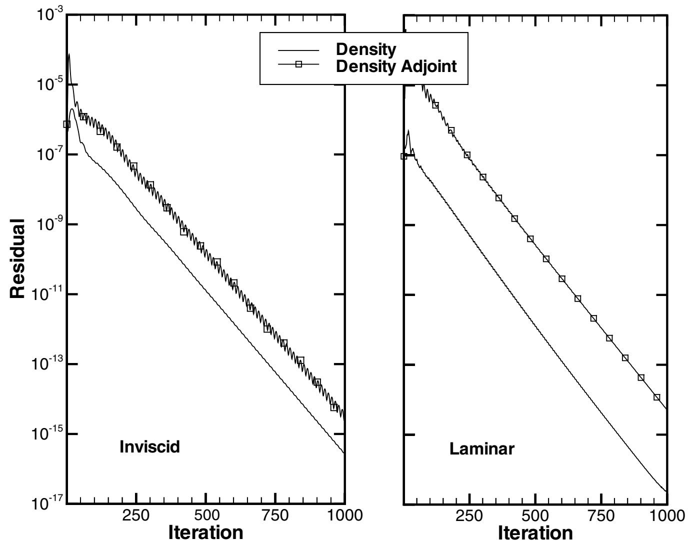

Fig. 9. Density residual histories for inviscid and laminar flow over ONERA M6 wing.

5.3. Turbulent wing

The final validation case is for turbulent flow over the ONERA M6 wing on the same grid that was used in the previous example. For this case, the freestream Mach number is 0.84, the angle of attack is 3.06°, and the Reynolds number is 5 million based on the mean aerodynamic chord. The convergence histories for the flow and adjoint equations shown in Fig. 10 exhibit similar asymptotic rates, where the convergence of the density equation eventually stalls as machine precision is achieved.

To further demonstrate the asymptotic nature of the flow and adjoint solution algorithms, the turbulent flow test case is also used to examine the convergence of a cost function and its derivatives. For this test, the wing has been parameterized by using the method of [55]. Fig. 11 shows the error in the lift coefficient as the flow solution converges; it also shows the error in the derivatives of lift with respect to freestream Mach number, angle of attack, and several shape design variables at the midspan location as the adjoint solution converges. For this case, the error is defined as the difference between the current value and the final converged result. The errors are reduced at a similar rate for each computation.

The turbulent flow test case is also used to evaluate the linearization storage strategies described above. Tests are performed on sixteen 250 MHz R10000 processors of a Silicon Graphics Origin 2000 system to simplify memory monitoring and to eliminate the effects of network traffic. The turbulent flow validation case has been repeated using each of the storage strategies; memory and wallclock statistics are shown in Table 1.

Recomputing all of the necessary linearizations for the adjoint residual yields a memory requirement roughly equivalent to that of the flow solution, while the wallclock time is approximately 17% less. The primary reason for this discrepancy in solution times can be attributed to the formation of the flux Jacobians for the left-hand side and their block decompositions. These

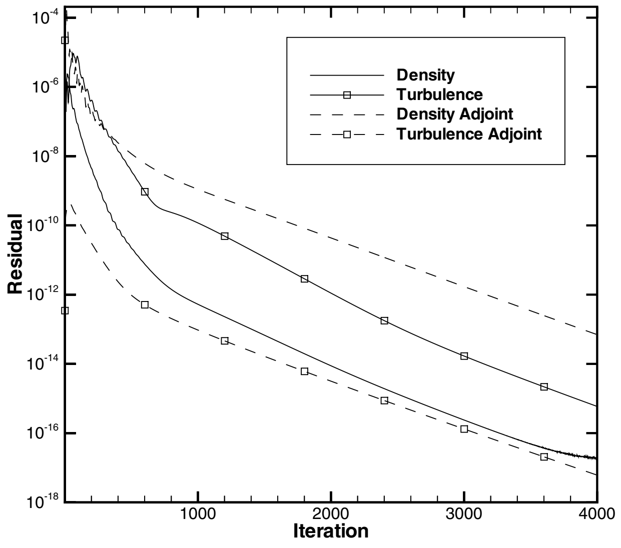

Fig. 10. Density and turbulence residual histories for turbulent flow over ONERA M6 wing.

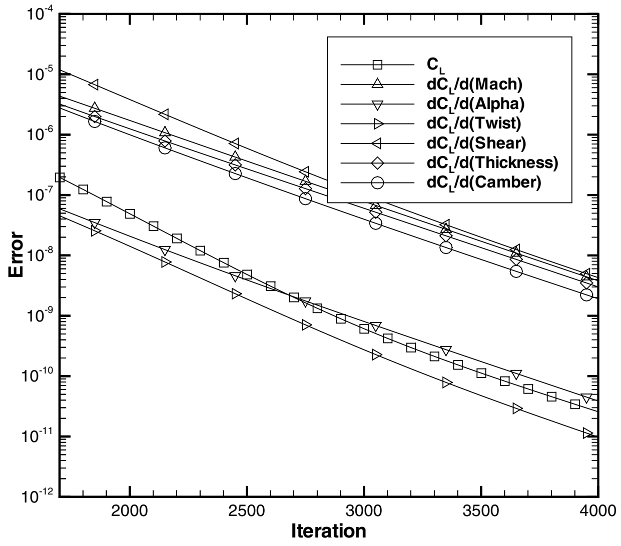

Fig. 11. Convergence of lift and lift derivatives for turbulent flow over ONERA M6 wing.

terms are evaluated at each time step during a flow solution, whereas they are constant for an adjoint computation and must only be formed once.

Alternatively, by storing the nearest-neighbor adjoint residual contributions that arise from the linearization of the viscous and turbulent terms, the wallclock time for an adjoint solution can be reduced to approximately half that of the flow solution while doubling the memory requirement. Finally, by storing the complete linearization of the second-order residual, the computational time

Table 1
Memory and CPU requirements for flow solver and various linearization storage strategies used for computing adjoint residual

Solver	Memory (GB)	Wallclock time (h)
Flow solver	1.5	50.4
Adjoint solver		
Explicit computation of all residual terms	1.6	42.0
Explicit computation of inviscid residual terms; viscous, turbulent residual terms stored	3.1	25.9
All residual terms stored	10.8	17.1

Table 2
Relative execution times for simultaneous computation of multiple adjoint solutions

Output functions	Relative execution time
C_L	1.00
C_L, C_D	1.44
C_{L_p}, C_{D_p}, C_{L_v}, C_{D_v}	2.37
C_{L_p}, C_{D_p}, $(C_{M_y})_p$, C_{L_v}, C_{D_v}, $(C_{M_y})_v$	3.31

is reduced to roughly one-third of the baseline flow solution; however, the memory requirement for this approach is a factor of 6 higher than the baseline flow solution algorithm. Based on these results, the viscous and turbulent terms are stored throughout the remainder of the current work, and the inviscid contributions are recomputed as needed.

The current turbulent flow test case is also used to demonstrate the relative efficiency in computing simultaneous adjoint solutions for several output functions. To specify an output function, the present implementation relies on a data structure that is identical to that used to store the outputs generated by the flow solver. In this manner, it is trivial to specify output functions for a subsequent adjoint computation on any arbitrary boundary or group of boundaries in the domain. For this test, the total lift, drag, and pitching moment coefficients are used as output functions, as well as the individual contributions from pressure and viscous forces. Relative timings based on the single-output function case are shown in Table 2, where simultaneous adjoint solutions for up to six output functions are obtained at approximately 45% of the cost of a separate additional computation.

6. Large-scale test cases

Two large-scale test cases are used to evaluate the solution algorithm on realistic configurations. Each of the grids has been generated by using the method described in [56]. In the interest of comparing asymptotic convergence behavior, solutions have been converged considerably beyond the usual requirements for engineering-level answers; the stated run times do not represent time required to obtain a sufficiently accurate solution.

6.1. High-lift configuration

Turbulent flow over the high-lift configuration [57,58] shown in Fig. 12 is computed using eighteen 1.7 GHz Pentium IV processors. The grid contains 846,863 nodes and 4,879,086 cells, and the implicit line construction algorithm places 418,523 nodes in the line-implicit region. Although only a single plane is shown, symmetry plane boundary conditions are used on both sides of the configuration. The freestream Mach number is 0.2, the angle of attack is 12°, and the Reynolds number is 9 million based on the stowed chord length. The CFL numbers for the flow equations and turbulence model are linearly ramped from 10 to 200 and 1 to 20, respectively, over the first 200 iterations. This combination of parameters is not necessarily optimal, but has been found to work well for a number of cases. For this computation, the objective function is based on the lift coefficient. The flow solver requires approximately 3.5 GB of memory and 36.2 h of wallclock time for the current example; the adjoint solver requires roughly 7.1 GB and 19.2 h. The

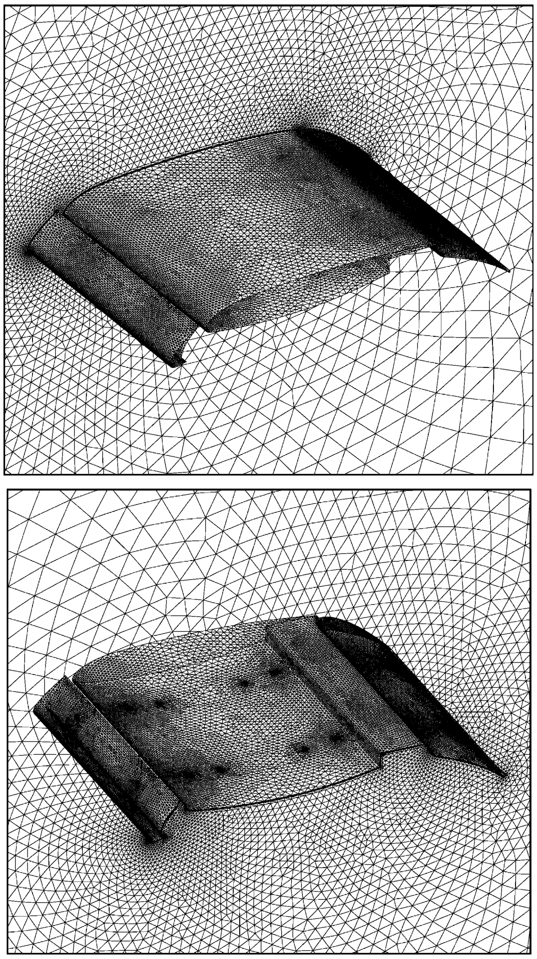
Fig. 12. Surface grid for high-lift configuration.

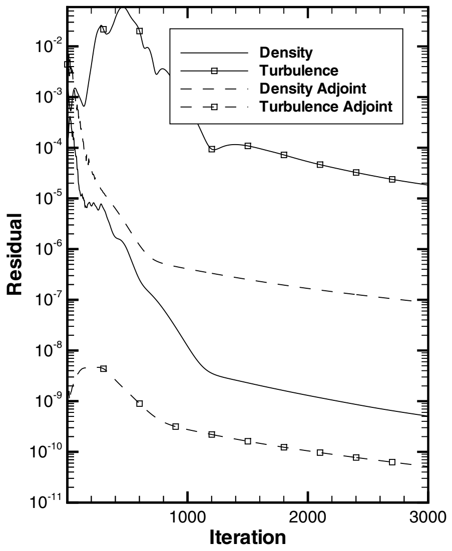

Fig. 13. Density and turbulence residual histories for turbulent flow over high-lift configuration.

convergence histories for the flow and adjoint solutions are shown in Fig. 13; the asymptotic rates are similar.

6.2. Modern transport configuration

For this test, transonic flow over the modern transport configuration shown in Fig. 14 is considered. The grid for this case contains 1,731,262 nodes and 10,197,838 cells. The implicit line

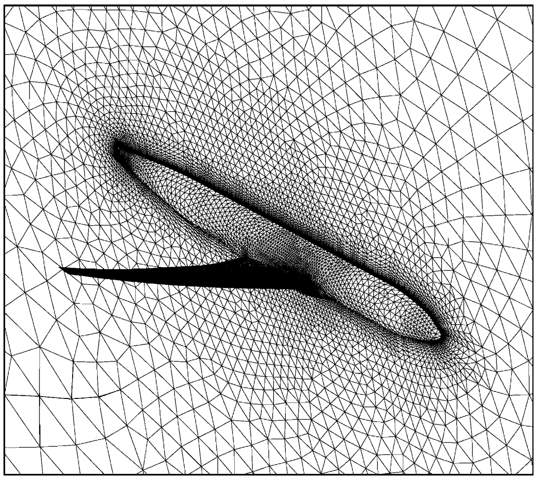

Fig. 14. Surface grid for modern transport configuration.

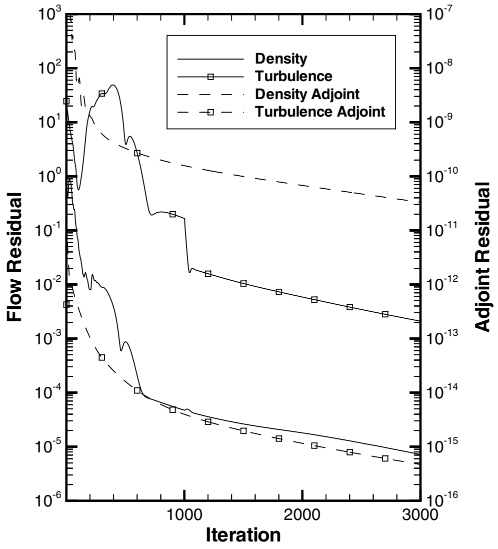

Fig. 15. Density and turbulence residual histories for turbulent flow over modern transport configuration.

construction algorithm places 1,220,567 nodes in the line-relaxation region. The freestream Mach number for this case is 0.84, the angle of attack is 2.25°, and the Reynolds number is 3 million based on the mean aerodynamic chord. The computations shown here are performed on twenty-three 2.4 GHz Pentium IV processors, and the cost function is based on the drag coefficient.

The convergence histories for the flow and adjoint solutions are shown in Fig. 15. The CFL numbers for the flow equations and turbulence model are linearly ramped from 10 to 200 and 1 to 20, respectively, over the first 200 iterations. The CFL number used for the turbulence model is increased to 200 after the first 1000 iterations. As shown in Fig. 15, the asymptotic rates for both computations are similar. For this example, the flow solution requires 7.4 GB of memory and 41.8 h of wallclock time; the adjoint solution requires 15.0 GB of storage and 32 wallclock hours.

7. Summary and future work

An adjoint solution algorithm that preserves discrete duality for turbulent flows on unstructured grids has been developed and implemented. The scheme is based on a backward-Euler discretization of the Reynolds-averaged Navier–Stokes equations with a tightly coupled one-equation turbulence model, where the linear problem at each time step is solved with a line-implicit relaxation in the boundary layer and a point-implicit technique through the remainder of the domain.

Results for several cases show that the exact dual scheme achieves asymptotic convergence behavior equivalent to that of the flow problem. By storing the viscous and turbulent contributions to the adjoint residual, the memory required for the algorithm is approximately twice that of the flow solution, but with execution times roughly 25–50% less for an equivalent number of iterations. For multiple output functions, a significant savings is achieved by simultaneously computing adjoint solutions within the context of a single execution, with a negligible amount of additional memory required.

Efforts are currently underway to develop a multigrid implementation for improved convergence rates and to extend the discretization to include mixed-element grids. Real gas physical models from existing hypersonic structured-grid solvers are also being added, with the long-term goal of achieving a self-adaptive analysis and design capability valid across the speed range [59].

Acknowledgements

The authors wish to thank Drs. James Thomas, Kyle Anderson, Steve Allmaras, and Peter Gnoffo for many helpful discussions pertaining to the current work. Dr. Jamshid Samareh and Elizabeth Lee-Rausch are gratefully acknowledged for providing the geometric parameterization and high-lift grid, respectively. The study would not have been possible without the computational resources provided by Sally Viken and Bil Kleb, and Drs. Mark Carpenter, Veer Vatsa, and Dick Wilmoth.

Appendix A. Derivation of exact dual outer iteration

In this appendix, the expression given by Eq. (15) is manipulated to give the exact dual iterative scheme as shown in Eq. (17). The derivation shown here is identical to that of [20].

Using the discrete form of integration by parts, namely

$$\sum_{n=0}^{N-1} a^{n+1}(b^{n+1} - b^n) = a^N b^N - a^0 b^0 - \sum_{n=0}^{N-1}(a^{n+1} - a^n) b^n. \tag{A.1}$$

Eq. (15) can be expanded as

$$\begin{aligned} \frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N = {} & \frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N - (\lambda^N)^{\mathrm{T}}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N + (\lambda^0)^{\mathrm{T}}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^0 + \sum_{n=0}^{N-1}\left\{(\lambda^{n+1} - \lambda^n)^{\mathrm{T}}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n \right. \\ & \left. + (\lambda^{n+1})^{\mathrm{T}}\boldsymbol{P}\left[\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n + \frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}}\right]\right\}. \end{aligned} \tag{A.2}$$

Rearranging terms and applying the initial condition $[\partial \boldsymbol{Q}^*/\partial \boldsymbol{D}]^0 = 0$ gives

$$\begin{aligned} \frac{\partial f}{\partial \boldsymbol{Q}^*}\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}^*}\right)^N = {} & \left[\frac{\partial f}{\partial \boldsymbol{Q}^*} - (\lambda^N)^{\mathrm{T}}\right]\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^N + \sum_{n=0}^{N-1}\left\{\left[(\lambda^{n+1} - \lambda^n)^{\mathrm{T}} + \left(\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{Q}^*}\right)^{\mathrm{T}}\boldsymbol{P}^{\mathrm{T}}\lambda^{n+1}\right]\left(\frac{\partial \boldsymbol{Q}^*}{\partial \boldsymbol{D}}\right)^n \right. \\ & \left. - \boldsymbol{P}^{\mathrm{T}}\lambda^{n+1}\frac{\partial \boldsymbol{R}_2}{\partial \boldsymbol{D}}\right\}. \end{aligned} \tag{A.3}$$

By applying the condition $\lambda^N = (\partial f/\partial \boldsymbol{Q}^*)^{\mathrm{T}}$ and the variable substitution shown in Eq. (16), the exact dual iteration takes the form of Eq. (17) through the elimination of the terms involving $\partial \boldsymbol{Q}^*/\partial \boldsymbol{D}$.

References

[1] Nielsen EJ. Aerodynamic design sensitivities on an unstructured mesh using the Navier–Stokes equations and a discrete adjoint formulation. PhD dissertation, Department of Aerospace and Ocean Engineering, Virginia Polytechnic Institute and State University, December 1998.

[2] Anderson WK, Bonhaus DL. Airfoil design on unstructured grids for turbulent flows. AIAA J 1999;37(2):185–91.

[3] Anderson WK, Venkatakrishnan V. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. Comput Fluids 1999;28(4):443–80.

[4] Nielsen EJ, Anderson WK. Aerodynamic design optimization on unstructured meshes using the Navier–Stokes equations. AIAA J 1999;37(11):1411–9.

[5] Nielsen EJ, Anderson WK. Recent improvements in aerodynamic design optimization on unstructured meshes. AIAA J 2002;40(6):1155–63.

[6] Jameson A. Aerodynamic design via control theory. J Sci Comput 1988;3:233–60.

[7] Jameson A, Pierce NA, Martinelli L. Optimum aerodynamic design using the Navier–Stokes equations. AIAA Paper 97-0101, January 1997.

[8] Reuther JJ, Jameson A, Alonso JJ, Rimlinger MJ, Saunders D. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers. J Aircraft 1999;36(1):51–60.

[9] Elliott J, Peraire J. Practical three-dimensional aerodynamic design and optimization using unstructured meshes. AIAA J 1997;35(9):1479–85.

[10] Soemarwoto B. Multipoint aerodynamic design by optimization. PhD dissertation, Department of Theoretical Aerodynamics, Delft University of Technology, December 1996.

[11] Mohammadi B. Optimal shape design, reverse mode of automatic differentiation and turbulence. AIAA Paper 97-0099, January 1997.

[12] Nemec M, Zingg DW. Towards efficient aerodynamic shape optimization based on the Navier–Stokes equations. AIAA Paper 2001-2532, 2001.

[13] Kim H-J, Sasaki D, Obayashi S, Nakahashi K. Aerodynamic optimization of supersonic transport wing using unstructured adjoint method. AIAA J 2001;39(6):1011–20.

[14] Soto O, Lohner R. A mixed adjoint formulation for incompressible turbulent problems. AIAA Paper 2002-0451, 2002.

[15] Sung C, Kwon JH. Aerodynamic design optimization using the Navier–Stokes and adjoint equations. AIAA Paper 2001-0266, 2001.

[16] Kim CS, Kim C, Rho OH. Sensitivity analysis for the Navier–Stokes equations with two-equation turbulence models. AIAA J 2001;39(5):838–45.

[17] Iollo A, Salas MD, Ta'asan S. Shape optimization governed by the Euler equations using an adjoint method. ICASE Report No. 93-78, November 1993.

[18] Newman III JC, Taylor III AC, Burgreen GW. An unstructured grid approach to sensitivity analysis and shape optimization using the Euler equations. AIAA Paper 95-1646, 1995.

[19] Giles MB, Pierce NA. An introduction to the adjoint approach to design. Flow, Turbulence Combust 2000;65(3 and 4):393–415.

[20] Giles MB. On the use of Runge–Kutta time-marching and multigrid for the solution of steady adjoint equations. AD2000 Conference in Nice, June 19–23, 2000.

[21] Giles MB. Adjoint code developments using the exact discrete approach. AIAA Paper 2001-2596, 2001.

[22] Giles MB. On the iterative solution of adjoint equations. In: Corliss G, Faure C, Griewank A, Hascoet L, Naumann U, editors. Automatic differentiation: from simulation to optimization. Springer-Verlag; 2001.

[23] Giles MB, Duta MC, Muller J-D, Pierce NA. Algorithm developments for discrete adjoint methods. AIAA J 2003;41(2):198–205.

[24] Monk P, Suli E. The adaptive computation of far-field patterns by a posteriori error estimation of linear functionals. SIAM J Numer Anal 1998;8:251–74.

[25] Paraschivoiu M, Peraire J, Patera A. A posteriori finite element bounds for linear-functional outputs of elliptic partial differential equations. Comput Methods Appl Mech Eng 1997;150:289–312.

[26] Pierce NA, Giles MB. Adjoint recovery of superconvergent functionals from PDE approximations. SIAM Rev 2000;42(2):247–64.
[27] Venditti DA, Darmofal DL. Adjoint error estimation and grid adaptation for functional outputs: application to quasi-one-dimensional flow. J Comput Phys 2000;164:204–27.
[28] Venditti DA, Darmofal DL. Grid adaptation for functional outputs: application to two-dimensional inviscid flows. J Comput Phys 2002;176:40–69.
[29] Muller JD, Giles MB. Solution adaptive mesh refinement using adjoint error analysis. AIAA Paper 2001-2550, 2001.
[30] Venditti DA. Grid adaptation for functional outputs of compressible flow simulations. PhD dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, June 2002.
[31] Park MA. Adjoint-based, three-dimensional error prediction and grid adaptation. AIAA Paper 2002-3286, 2002.
[32] Venditti DA, Darmofal DL. Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows. J Comput Phys 2003;187:22–46.
[33] Peraire J, Vahdati M, Morgan K, Zienkiewicz OC. Adaptive remeshing for compressible flow computations. J Comput Phys 1987;72:249–66.
[34] Pirzadeh SZ. A solution-adaptive unstructured grid method by grid subdivision and local remeshing. J Aircraft 2000;37(5):818–24.
[35] Park MT, Kwon OJ. Unsteady flow computations using a 3-D parallel unstructured dynamic mesh adaptation algorithm. AIAA Paper 2001-0865, 2001.
[36] Warren GP, Anderson WK, Thomas JL, Krist SL. Grid convergence for adaptive methods. AIAA Paper 91-1592, 1991.
[37] Elliott J. Discrete adjoint analysis and optimization with overset grid modelling of the compressible high-Re Navier–Stokes equations. In: 6th Overset Grid and Solution Technology Symposium, Fort Walton Beach, FL, October 2002.
[38] Anderson WK, Bonhaus DL. An implicit upwind algorithm for computing turbulent flows on unstructured grids. Comput Fluids 1994;23(1):1–21.
[39] Anderson WK, Rausch RD, Bonhaus DL. Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids. J Comput Phys 1996;128:391–408.
[40] Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 1986;7(3):856–69.
[41] Spalart PR, Allmaras SR. A one-equation turbulence model for aerodynamic flows. AIAA Paper 92-0429, 1992.
[42] Roe PL. Approximate Riemann solvers, parameter vectors, and difference schemes. J Comput Phys 1981; 43(2):357–72.
[43] Van Leer B. Flux vector splitting for the Euler equations. Lect Notes Phys 1982;170:501–12.
[44] Baldwin BS, Barth TJ. A one-equation turbulence transport model for high Reynolds number wall bounded flows. NASA Technical Memorandum 102847, August 1991.
[45] Allmaras SR. Multigrid for the 2-D compressible Navier–Stokes equations. AIAA Paper 99-3336, 1999.
[46] Schmitt V, Charpin F. Pressure distributions on the ONERA-M6 wing at transonic Mach numbers, experimental database for computer program assessment. AGARD-AR-138, May 1979. p. B1-1–44.
[47] Trottenberg U, Oosterlee C, Schuller A. Multigrid. San Diego: Academic Press; 2001. p. 134.
[48] Venkatakrishnan V. Improved multigrid performance of compressible Navier–Stokes solvers. AIAA Paper 98-2967, 1998.
[49] Mavriplis D. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. J Comput Phys 1998;145(1):141–65.
[50] Redeker G. DLR-F4 wing body configuration. AGARD-AR-303, vol. 2, August 1994.
[51] Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 1998;20(1):359–92.
[52] Karypis G, Kumar V. Multilevel algorithms for multi-constraint graph partitioning. University of Minnesota Technical Report No. 98-019, 1998.
[53] Levy DW, Zickuhr T, Vassberg J, Agrawal S, Wahls RA, Pirzadeh S, et al. Summary of data from the first AIAA CFD drag prediction workshop. AIAA Paper 2002-0841, 2002.

[54] Kleb WL, Nielsen EJ, Gnoffo PA, Park MA, Wood WA. Collaborative software development in support of Fast Adaptive AeroSpace Tools (FAAST). AIAA Paper 2003-3978, 2003.
[55] Samareh JA. A novel shape parameterization approach. NASA TM-1999-209116, May 1999.
[56] Pirzadeh S. Three-dimensional unstructured viscous grids by the advancing-layers method. AIAA J 1996;34(1):43–9.
[57] Spaid FW. High Reynolds number multielement airfoil flowfield measurements. J Aircraft 2000;37(3):499–507.
[58] Rumsey CL, Lee-Rausch EM, Watson RD. Three-dimensional effects on multi-element high lift computations. AIAA Paper 2002-0845, 2002.
[59] Alexandrov N, Alter S, Atkins H, Bey K, Bibb K, Biedron R, et al. Opportunities for breakthroughs in large-scale computational simulation and design. NASA TM-2002-211747, April 2002.